02

Primer filtro

Transcripción

[00:00] Hola a todos y todas, bienvenidos y bienvenidas a este nuevo video sobre Servlet. En el video pasado les dije que íbamos a ver dónde colocar esta parte de este código del inicio de sesión, ya que no parece una buena idea colocarlo acá dentro del controlador, porque nuestro controlador está comenzando a tener varias funciones, por lo menos dos, y no es una buena idea.

[00:25] El controlador, la idea original del controlador era saber a cuál acción nosotros tenemos que ir, cuál acción tenemos que ejecutar dada una cierta request y le hemos agregado acá una parte de seguridad. Entonces le hemos agregado, esto. Imaginen, por ejemplo, que su jefe o mi jefe en el trabajo me dice: “Bruno, estamos con la aplicación y esta aplicación está muy lenta, realmente necesito que averigües qué es lo que está ocurriendo”.

[00:59] ¿Qué es lo que está ocurriendo en una aplicación que está muy lenta? Puede ser debido a muchas muchas cosas. Por ejemplo, puede ser que la internet de nuestro jefe esté lenta, puede ser que la computadora de nuestro jefe esté lenta, puede ser que la base de datos esté ejecutándose de una forma lenta o puede ser, por ejemplo, nuestro código también que esté lento, realmente hemos aplicado algunos alguna cosa así que lo hace lento.

[01:21] ¿Entonces cómo sabríamos si nuestra aplicación está lenta o no, si la culpa es realmente de nuestro código? Bueno, por ejemplo, iríamos a nuestra listaEmpresas. Supongamos que es acá donde nuestro jefe nos dice que está

lenta la aplicación, y podemos contar el tiempo entre que se inicia y que finaliza esa acción, ¿cómo haríamos eso?

[01:46] Tenemos un recurso acá en system.currentTimeMillis(); esa función lo que nos trae es la cantidad de milisegundos que han ocurrido desde el año 1970. 1970 se considera el año cero dentro de la computación. Números positivos van a ser después de 1970, números negativos, antes.

[02:07] Entonces, cuando yo entro y coloco currentTimeMillis me da el momento exacto en el que pasó por esta línea, me dice cuántos milisegundos han pasado desde 1970. Entonces, nosotros, por ejemplo, lo podemos guardar en una variable del tipo long, que vamos a llamarla antes, porque es antes de ejecutar nuestro código, y podríamos tener una variable después.

[02:34] Voy a copiar este código, "Ctrl + C", "Ctrl + V". Y voy a cambiar acá a después, bien. "Ctrl + S". Entonces tenemos un antes y un después. ¿Qué es lo que haríamos para poder saber cuántos milisegundos han pasado desde este momento hasta este momento? Podemos hacer un system.out.println y vamos a colocar en “Tiempo de ejecución +()”. Y acá vamos de hacer la cuenta de después menos antes.

[03:16] Con esto calculamos cuánto tiempo ha pasado entre un momento y el otro. Vamos a probarlo. Voy a ir a nuestro Google Chrome y no sé si se inició el servidor, no, lo había parado. Voy a guardar y vamos a iniciar el servidor y vamos a ver, ahora sí, sí vamos a nuestra entrada, nuestra acción loginForm. Enter, bien. Vamos a ver acá en nuestra consola a ver si se ejecutó nuestro código.

[03:50] ¿Puede ser que no? Ah, claro, es que lo puse en listaEmpresas, entonces vamos a loguearnos, vamos a hacer un Bruno 12345. Enviar. Ahora sí. Está, sí, sí. Ahora sí, tiempo de ejecución 0 milisegundos. No tardó nada en hacer eso, posiblemente es porque estaba en caso. Vamos a hacer un F5 acá solo por las dudas.

[04:15] “Ctrl + F5” para eliminar el catch. Bueno, aparece 0 segundos, 0 milisegundos. La cuestión es que con esto, por ejemplo, estamos viendo cuánto tiempo demora listaEmpresas, pero si nosotros queremos saber cuánto demora nuestro eliminarEmpresa o nuestro modificarEmpresa, nuestra nuevaEmpresa, entonces tenemos que hacer "Ctrl + C", "Ctrl + V" de todo este código.

[04:40] No tiene mucho sentido, ya dijimos que no es una buena idea ir desparramando todo este código de un lugar para otro en cada parte de nuestro código. ¿Y qué ocurre por ejemplo, también si no solamente nos piden eso, nos dice que está muy lento, sino que nos dicen, Bruno, queremos saber quién hizo cada una de las acciones?

[04:59] ¿Quién es el que creó nueva empresa, quién es el que eliminó una empresa? ¿Cuál usuario hizo eso? Sería otro código que va a desparramarse a lo largo de todos nuestros archivos que queramos, por ejemplo, hacer esa auditoría.

[05:15] Entonces tendríamos algo por el estilo, tendríamos en nuestra listaEmpresas un código sobre auditoría, un código por ejemplo sobre seguridad, un tryCatch, un código try, intentar esto, catch, ver cuál es el error y lo mismo en cada una de esas otras acciones, en mostrarEmpresa, modificarEmpresa, etcétera.

[05:38] Entonces tendríamos ese mismo código en cada una de estas acciones, y no es una buena idea realmente. ¿Entonces qué es lo que tendríamos que hacer? Por ejemplo, nosotros pensaríamos: “No lo vamos a colocar en cada una de las acciones, entonces coloquémoslo en el controlador”. Bueno, pero el controlador entonces perdería un foco.

[06:01] Ya el controlador ya no sería ver a cuál acción tengo que llamar, sino que ya sería una mezcla de varias cosas y eso sería muy difícil de después poder hacer un mantenimiento si queremos cambiar en alguna cosa sobre la

auditoría, por ejemplo, tenemos que revisar en cada línea del código a ver dónde es que tenemos que hacer ese cambio, si es que algún problema tiene la auditoría, por ejemplo.

[06:26] Podría ser que el problema no está en la auditoría, entonces tenemos que probar varias cosas. Es realmente un infierno hacer eso. Entonces lo que nos ofrece Tomcat y mundo Servlet es lo que llamamos de filtro, ese filtro que vemos acá, un filtro, es una cosa muy, muy parecida a un Server, ya van a ver la implementación, pero es algo que ocurre antes de poder llegar a nuestro controlador.

[06:56] Entonces es una pieza de código que nos va a filtrar, como dice el nombre, ver si una petición cumple con algunos requisitos. Si cumple, entonces pasa por el controlador, por ejemplo, puede pasar, si no cumple entonces directamente se filtra, se para esa ejecución.

[07:17] Haciendo una analogía, habíamos dicho que nuestro controlador es como una puerta para nuestra casa, pero ahora pensemos que antes de esa puerta nosotros tenemos un pasillo donde tenemos una persona que está vigilando por ejemplo y que está viendo a ver si la persona que va a entrar realmente, por ejemplo, en un banco no tiene ningún objeto sospechoso.

[07:40] Si no tiene nada sospechoso, entonces puede entrar y continuar. Sería más o menos eso, la idea del filtro. Si la persona tiene una cosa sospechosa, no puede entrar al banco directamente. Entonces vamos a intentar hacer, vamos a no intentar no. Vamos a crear ese filtro.

[07:57] Vamos a ir a acá en nuestra listaEmpresas, voy a hacer un "Ctrl + S" y dentro de nuestra carpeta de servlets, de nuestro paquete servlet vamos a hacer un clic derecho new class. Tenemos acá también que podemos crear un filtro directamente, pero eso lo vamos a ver después, una vez de que ya hemos creado esa clase y lo vamos a ver bien cómo es que se crea este filtro.

[08:26] Entonces, vamos a hacer clic en class. ¿Cómo les llamamos? Voy a llamarlo de MonitoreoFilter. Perfecto, finish. Bien. Hemos creado esa clase y esta clase yo les dije que es muy, muy parecido a un servlet. Veamos nuestro servlet qué es lo que tiene. Nuestro servlet extiende de un httpServer.

[08:54] Entonces acá nuestro monitoreo no va a extender de una clase, pero sí va a implementar una interfaz, va a cumplir un contrato de un filtro, todo filtro tiene que tener ese contrato, esos métodos. Entonces vamos a hacer eso. Implements Filter. Y acá en ese filter tenga cuidado al importar, porque hay varios filtros, por ejemplo, Java tiene su filtro, que significa otra cosa que el filtro de los servlets.

[09:27] Que por ejemplo Spring tiene también sus filtros, que es diferente, entonces tengan cuidado al importar. En este caso yo voy a usar el jakarta servlet, que es el filtro de servlets. Y vean que nos está pidiendo ahora que tenemos que implementar métodos. Vamos a ir acá al foquito, add unimplemented methods. Bien.

[09:52] Perfecto, acá nos trae un método un doFilter. Fíjense que también es parecido con nuestro servlet que nosotros en los servers teníamos el doPost, doGet y también tenemos este servlet, en este caso nosotros tenemos un doFilter, bien parecido y no acá va a ir lo que es el parecido, sino que por ejemplo también tenemos nuestro ServletRequest y ServletResponse.

[10:18] Y también tenemos un filterChain, ese es ya lo vamos a ver, entonces acá tenemos un request, acá tenemos un response. Fíjense que en nuestra única entrada a servlet, nosotros teníamos un httpServletRequest y un httpServletResponse. Fíjese que ellos por ejemplo, extienden de servletRequest, entonces nuestro Servlet es algo más específico, mientras que nuestro filtro tiene estos elementos, ServletRequest y ServletResponse, que son más genéricos pero que vienen todos del mismo lugar.

[11:02] Entonces tenemos nuestro request, tenemos nuestro response y tenemos ese FilterChain que bueno, yo le voy a llamar de chain, que significa filter chain. Chain significa cadena. Entonces, ¿qué es ese chain? Ya lo vamos a ver. Antes que todo, ya que yo les dije que este filtro bien es lo que estamos implementando, acá es donde vamos a hacer el inicio de nuestro contador de tiempo y vamos a finalizarlo acá.

[11:37] Vamos a ir, vamos a copiar, mejor que copiar vamos a cortar lo que hemos creado de nuestra listaEmpresas, este long de antes, "Ctrl + X", voy a sacar esto. Voy a pegarlo acá. Bien. Y después de nuestra listaEmpresas voy a cortar este long que había puesto acá de después, pegar. ¿Y qué es lo que vamos a tener? Nosotros tenemos el control de lo que es antes y lo que es después, antes y después.

[12:11] Pero nos falta ahora poder llamar realmente a nuestro controlador y que bueno, de nuestro controlador haga todo lo que tiene que hacer y el modelo, todo eso. Pero a nosotros nos falta llamar a ese controlador. ¿Cómo hacemos para llamar ese controlador? Bueno, ese mismo chain que estábamos viendo acá es él, esa cadena es la que nos permite después llamar a ese controlador.

[12:41] ¿Cómo usamos esa cadena? Acá va a ir esa ejecución del controlador. Y entonces nosotros vamos a hacer un chain. y aquí tenemos ese método doFilter. Ese doFilter es el que va a llevar nuestra request y nuestra response. Esta request que tenemos acá y esa response la vamos a reenviar a lo que sigue de la cadena.

[13:12] ¿Cómo es que sabe a dónde tiene que ir? Bueno, es bastante fácil. Como les dije, nuestro filtro es muy parecido a nuestro servlet. Nuestro servlet acá tenemos nuestro mapeamiento, ese webServlet que nos dice cuando la persona va a /entrada, entonces entra por acá. Nosotros vamos a tener algo igual dentro o muy parecido, en realidad.

[13:39] Dentro de nuestro filtro nosotros vamos a tener un @WebFilter. Ese @WebFilter tiene dentro también como teníamos ese urlPatterns, que no lo colocamos acá, voy a colocarlo dentro de nuestra entradaServlet, ese urlPatterns = "/entrada". Nosotros también vamos a tener ese urlPatterns.

[14:03] Entonces (urlPatterns = "/entrada"), "Ctrl + S". Entonces, ¿cómo es que nuestro servlet no se confunde si tenemos un urlPattern de entrada acá y tenemos un urlPattern de entrada acá? Son iguales. Bueno, ¿qué es lo que ocurre? Acá estamos usando un WebFilter. Automáticamente Tomcat sabe que nuestro filtro viene antes de llamar a nuestro servlet.

[14:31] Entonces va a ejecutarse antes este WebFilter de acá antes que nuestro WebServlet. Entonces eso, ya con eso ya nos da toda esa cadena de ejecución. De ahí viene la palabra chain, cadena de ejecución. Primero va a ir a un filtro y después continuamos esa cadena a nuestro Servlet.

[14:55] Y ya que estamos acá, ya que nosotros acá si conseguimos ver cuál es nuestra request, acá en la listaEmpresas, bueno, acá teníamos también nuestra red, pero acá me gustaría saber exactamente de cuál filtro, perdón, de cuál acción es la que vamos a medir, cuál acción es la que estamos midiendo en este momento, entonces acá en este system.out del tiempo de ejecución, quiero agregarle cuál es la acción que nosotros tenemos.

[15:23] Como les dije que esto es igual a un servlet, nosotros podemos también obtener nuestro param acá. Entonces request.getParametar, y acá, ¿cuál es el parámetro que definí cuál es la acción que nosotros tenemos que ejecutar? Acción. Acción es el parámetro, si nosotros nos vamos acá, nosotros obtenemos el parámetro acción.

[15:51] Y con eso es lo que sabemos en cuál acción nosotros vamos a ejecutar, entonces nosotros obtenemos acción y voy a colocarlo acá en un string acción = request.getParameter. Entonces esa acción, nosotros ya podemos usarlo acá

en nuestro system.out. Entonces voy a colocar “parámetro de ejecución de la acción :” +acción+.

[16:22] Y acá le quiero agregar una cosa linda, solo para que se vea mejor, una flechita, entonces nos va a decir cuál es la acción y cuánto tiempo ha demorado. Vamos a ejecutar eso, voy a dar un run en nuestro server. Bien y vamos a salir, voy a colocar Bruno 12345. Enviar. Perfecto. Y vamos a ver acá, okay.

[16:53] Tenemos acá tiempos de ejecución de cada cosa. Tenemos nuestro logout 7 milisegundos, loginForm 213 milisegundos, nuestro Login demoró 3 milisegundos y nuestra listaEmpresas 182 milisegundos. Entonces vean que colocamos el código en un único lugar pero nos está sirviendo para ver todas nuestras acciones. Miren qué locos. Miren, qué potente esto de filtros.

[17:19] Y por ejemplo, si el día de mañana nosotros decimos: “No quiero más ejecutar ese filtro de ver cuánto tiempo demora cada cosa”, perfecto. Comentamos acá nuestro webFilter y listo, ya no funciona más, modificamos una línea y ya modificamos completamente el comportamiento de nuestro sistema. ¿Vieron?

[17:38] Así como por ejemplo, también podemos colocar cosas más inteligentes como decir: “Ah, bueno, solamente si estoy intentando ejecutar la acción mostrarEmpresas”, entonces ahí quiero medir cuál es nuestro tiempo de ejecución, por ejemplo. Entonces con esto nosotros hemos visto ya esta potente herramienta que tenemos de filtros y la idea ya a ser que este código d